Journal of Machine Learning Research 25 (2024) 1-6 Submitted 9/22; Revised 6/24; Published 7/24

PAMI: An Open-Source Python Library for Pattern Mining

R. Uday Kiran UDAY.RAGE@GMAIL.COM
The University of Aizu
Aizu-Wakamatsu, Fukushima, 965-8580, Japan

P. Veena RAGE.VINNY@GMAIL.COM
The University of Aizu
Aizu-Wakamatsu, Fukushima, 965-8580, Japan

Masashi Toyoda TOYODA@TKL.IIS.U-TOKYO.AC.JP
Institute of Industrial Science, The University of Tokyo
Tokyo, 153-8505, Japan

Masaru Kitsuregawa KITSURE@TKL.IIS.U-TOKYO.AC.JP
Research Organization of Information and Systems,
Tokyo, 105-0001, Japan,
The University of Tokyo,
Tokyo, 113-8654, Japan

Editor: Sebastian Schelter

Abstract

Crucial information that can empower users with competitive information to achieve socio-economic development lies hidden in big data. **Pattern mining** aims to discover this needy information by finding user interest-based patterns in big data. Unfortunately, existing pattern mining libraries are limited to finding a few types of patterns in transactional and sequence databases. This paper tackles this problem by providing a cross-platform open-source Python library called PAttern MIning (PAMI). PAMI provides several algorithms to discover different types of patterns hidden in various types of databases across multiple computing architectures. PAMI also contains algorithms to generate various types of synthetic databases. PAMI offers a command line interface, Jupyter Notebook support, and easy maintenance through the Python Package Index. Furthermore, the source code is available under the GNU General Public License, version 3. Finally, PAMI offers several resources, such as a user's guide, a developer's guide, datasets, and a bug report.

Keywords: Big data, data science, data mining, machine learning, artificial intelligence, pattern mining, open-source

1. Introduction

Big data is ubiquitous. Useful information that can help end-users achieve socio-economic development lies hidden in this data. Pattern mining (Agrawal et al., 1994; Luna et al., 2019; Kaushik et al., 2023) is a key knowledge discovery technique that aims to find the needy information by discovering user interest-based patterns in big data. In the literature, researchers presented several open-source data mining libraries (e.g., WEKA (Witten et al., 2011), Mahout (Mahout, 2020), Knime (knime, 2022), RapidMiner (RapidMiner, 2022), MLxtend (Raschka, 2018), and Orange (Borondics, 2022)) to perform a wide range of data

mining techniques. Unfortunately, these generic libraries offer a limited set of pattern-mining algorithms that mainly focus on finding frequently occurring patterns in a binary transactional database. Specialized pattern mining libraries, Coron (Couceiro, 2022) and LUCS-KDD (Coenen, 2022), though offer a slightly more comprehensive choice of pattern mining algorithms, they are no longer in active development. The source code of Coron is not public, and one cannot use the LUCS-KDD source code for commercial purposes. SPMF (Fournier-Viger et al., 2014; Fournier-Viger et al., 2016) is another specialized pattern mining library that provides a wide range of pattern mining algorithms. Although this library is currently developing, it focuses mainly on developing sequential algorithms to find frequent patterns in transactional and sequence databases. Since the algorithms in SPMF are available in Java, its maintenance at the user-end is a bit difficult and, more importantly, challenging to integrate with the existing Python-based machine learning libraries, such as TensorFlow (Abadi et al., 2016) and Sklearn (Pedregosa et al., 2011). In this context, an open-source Python library, PAttern MIninig (PAMI), has been developed with a critical focus on seamless data communication across different machine learning libraries. PAMI provides over 70 different algorithms for pattern mining. These algorithms cover a wide range of patterns (e.g., frequent patterns, periodic-frequent patterns, high utility patterns, geo-referenced frequent patterns, uncertain-frequent patterns, and fuzzy-frequent patterns) found in various database types (e.g., transactional databases, temporal databases, utility databases, uncertain databases, and fuzzy databases) across the heterogeneous computing architectures (e.g., CPU-based, GPU-based, and parallel algorithms based on the map-reduce framework). Implementations of several of these algorithms are available only in PAMI. A distinctive feature of PAMI is that it offers over 20 algorithms to find periodic patterns in temporal, spatiotemporal, time series, fuzzy, and uncertain databases. To our knowledge, PAMI is the only dedicated Python library available for pattern mining purposes.

2. Architecture

PAMI is written in camel casing and follows a hierarchical structure. The first level is the library name, i.e., PAMI. The second level is the conceptual (or theoretical) model of a pattern (e.g., frequent, correlated, and periodic patterns). The third level is the type of pattern (e.g., all, maximal, closed, and top-k patterns) discovered from the data. The fourth level contains the names of mining algorithms. Thus, we can import an algorithm in PAMI using the following syntax: *import PAMI.theoreticalModelOfPattern.patternType.algorithmName*. For instance, a statement *import PAMI.frequentPatterns.basic.fpGrowth* imports the basic frequent pattern mining algorithm called FP-growth (Han et al., 2000). Besides mining algorithms, PAMI includes additional procedures for generating synthetic databases, converting data frames into various forms of databases, and visualizing patterns.

Currently, the algorithms provided in PAMI facilitate the users to discover 27 different types of interesting patterns (see Figure 1) in the following six types of databases:

1. **Transactional database:** Let $A = \{A_1, A_2, \cdots, A_n\}, n >= 1$, be a set of binary attributes (or items). Let $T_i \subseteq A,\ i \geq 1$, be a binary transaction. A transactional database, denoted as TDB, is an unordered collection of transactions. That is, $TDB = \cup_{k=1}^{m} T_k$.

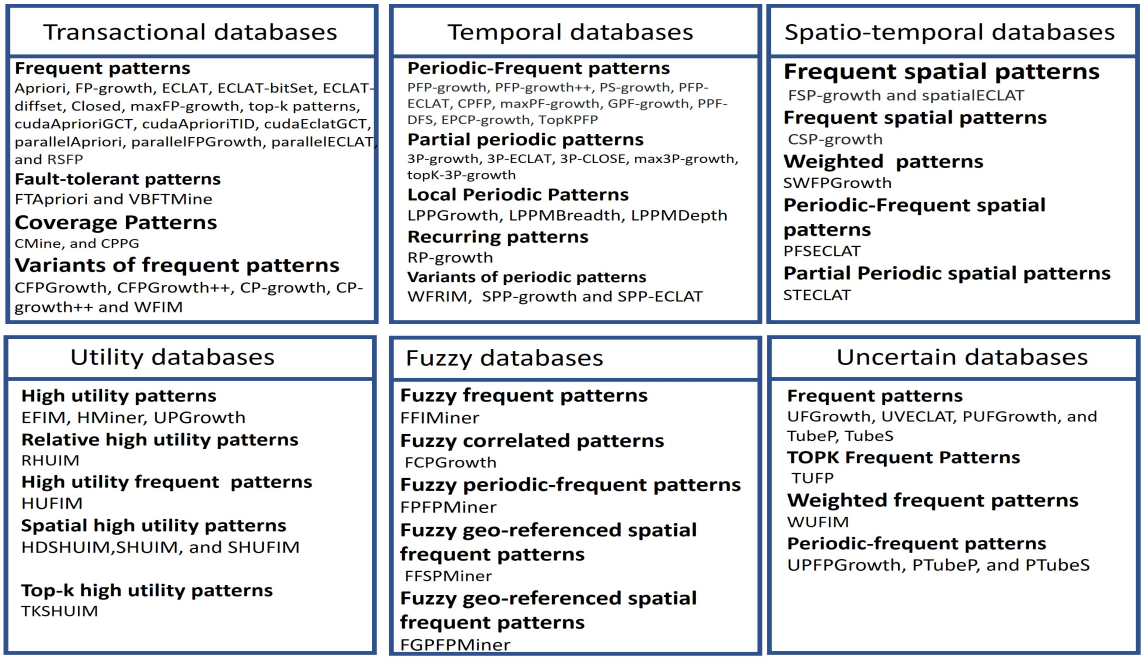

Figure 1: Different types of algorithms available in PAMI

2. **Temporal database:** A temporal transaction, T_i, is a pair containing a timestamp and a set of attributes. That is, $T_i = \{ts, \widehat{A}\}$, where $ts \in \mathbb{R}^+$ represents timestamp and $\widehat{A} \subseteq A$ is a set of items. A temporal database, denoted as $TempDB$, is an ordered collection of transactions. That is, $TempDB = \cup_{k=1}^{m} T_k$.

3. **Spatiotemporal database:** If every attribute in a temporal database carries spatial information, we call that database a spatiotemporal database.

4. **Utility database.** Let $UT_i = \cup_{p=1}^{n} VA_p$, where $VA_p \in \mathbb{R}^+$ represents the value of an attribute A_p in a transaction T_i. A utility database, denoted as $UTDB = \cup_{k=1}^{m} UT_i$. In other words, a utility database is a non-binary transactional database in which each attribute contains a real-valued number.

5. **Fuzzy database.** A fuzzy database, denoted as FDB, is generated by transforming a utility database using a set of fuzzy functions. That is, $FDB = F(UTDB)$, where $F(.)$ is a fuzzy function and $UTDB$ is a utility database.

6. **Uncertain database.** An uncertain database, denoted as $UCDB$, is a transactional database where every binary attribute in a transaction is associated with a probability value ranging between 0 and 1.

3. Using PAMI

PAMI is implemented in Python 3.6 and is cross-platform. The source code of PAMI is made available through GitHub. The execution code of PAMI is made available through

the Python Package Index (PYPI) so that users can easily install, update, or delete our library using the 'pip' command. Figure 2 presents the inputs and outputs by which the users can interact with a pattern mining algorithm available in PAMI.

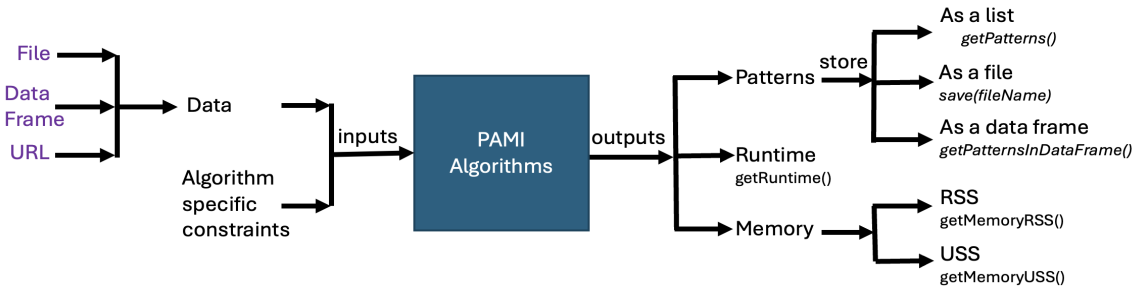

Figure 2: Inputs and outputs of a pattern mining algorithm in PAMI

The PAMI documentation and Jupyter Notebooks provide basic and advanced examples of executing an algorithm. The users can execute the algorithms in PAMI through a terminal, Jupyter Notebook, or an Integrated Developer Environment (IDE). The users can store the patterns an algorithm generates in a file or a data frame for further processing. Executing an algorithm is just a few lines of code. The source code of each algorithm also contains a sample program detailing how to execute the corresponding algorithm. For example, the following code runs the FP-growth algorithm on an input file with its minimum support parameter set to 0.1. (The T10I4D100K.dat file can be downloaded from `http://fimi.uantwerpen.be/data/`)

```
from PAMI.frequentPattern.basic import fpGrowth  as alg
#Initialize
obj = alg.fpGrowth(inputFile='T10I4D100K.dat',minSup=0.4)
#Mine
obj.mine()
#Save
obj.save('patterns.txt')
print('Runtime:'+str(obj.getRuntime()))
print('Memory:'+str(obj.getMemoryRSS()))
```

The users can easily integrate our source code into other Python programs as each algorithm exists in its sub-package. More importantly, using data frames in our library facilitates the pipelining with other machine learning libraries, such as TensorFlow and Sklearn. We also provide over 40 large databases (Kiran, 2022) that can be used with the algorithms offered in PAMI. Furthermore, this library can be beneficial for educational purposes (say, teaching data mining courses) or evaluating algorithms' performance. Finally, the website also provides information on how the output of various machine learning libraries can be pipelined into PAMI to discover needy information.

4. Conclusion

PAMI is a fast and comprehensive Python library for various pattern mining tasks. It contains over a hundred algorithms covering a broad spectrum of pattern mining tasks in heterogeneous computing environments. The algorithms in PAMI can be executed on a terminal, Integrated Developers Environment, or Jupyter Notebook. PAMI can seamlessly integrate with other machine learning libraries. Therefore, PAMI is a potentially indispensable toolbox for data mining and machine learning. Future versions of PAMI intend to support data streams, sequence data, and graphs.

Acknowledgement

We express our deepest gratitude to Miss Palla Likhitha for her unwavering dedication and support throughout the development of the PAMI library. We extend special recognition to Miss Eddula Raashika, Mr. Tarun Sreepada, Mr. Suzuki Shota, Miss Kattamuri Vanitha, and Miss Palla Madhavi for their meticulous efforts in designing and developing the sample documentation and test cases. Our heartfelt thanks go to all the students, collaborators, and users who contributed to the PAMI library's development. We also appreciate the anonymous reviewers for their insightful comments and suggestions on our manuscript.

We are grateful to GitHub (`https://github.com/`) for providing a robust platform for version control and collaboration, which has been crucial for managing our codebase and ensuring seamless teamwork and integrity throughout the development process.

We thank the Python Package Index (`https://pypi.org/`) for hosting essential PAMI Python packages. This repository has been invaluable in distributing and managing dependencies, facilitating our software's efficient deployment and utilization.

We acknowledge ReadtheDocs (`https://about.readthedocs.com/`) for offering an invaluable platform for hosting and generating documentation for PAMI. The documentation hosted on Read the Docs has greatly enhanced the accessibility and understanding of PAMI's functionalities for our users and contributors.

I, RAGE Uday Kiran, extend my heartfelt gratitude to my Ph.D. supervisor, Prof. P. Krishna Reddy, for his invaluable support and guidance throughout my academic journey. I also wish to thank Prof. Philippe Fournier-Viger for developing the SPMF software, which inspired me to design and develop the PAMI library. Special thanks to Mr. Koji Zettsu and Mr. Minh-Son Dao for their support in developing the fundamental technologies for the PAMI library.

Lastly, we acknowledge the University of Aizu's institutional support, which has significantly contributed to our research and development efforts in PAMI.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases*, volume 1215, pages 487–499, 1994.

Ferenc Borondics. Orange data mining. https://orangedatamining.com/, 2022. [Online; accessed 31-August-2022].

Frans Coenen. LUCS-KDD. https://cgi.csc.liv.ac.uk/~frans/KDD/Software/, 2022. [Online; accessed 31-August-2022].

Miguel Couceiro. Coron data mining. http://coron.loria.fr/site/index.php, 2022. [Online; accessed 31-August-2022].

Philippe Fournier-Viger, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Cheng-Wei Wu, and Vincent S. Tseng. Spmf: A java open-source pattern mining library. *J. Mach. Learn. Res.*, 15(1):3389–3393, jan 2014. ISSN 1532-4435.

Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. The SPMF open-source data mining library version 2. In *ECML PKDD, Part III*, volume 9853 of *Lecture Notes in Computer Science*, pages 36–40. Springer, 2016.

Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, may 2000. ISSN 0163-5808.

Minakshi Kaushik, Rahul Sharma, Iztok Fister Jr. au2, and Dirk Draheim. Numerical association rule mining: A systematic literature review, 2023.

RAGE Uday Kiran. Datasets for pattern mining. https://u-aizu.ac.jp/~udayrage/datasets.html, 2022. [Online; accessed 31-August-2022].

knime. software. https://www.knime.com/, 2022. [Online; accessed 31-August-2022].

José María Luna, Philippe Fournier-Viger, and Sebastián Ventura. Frequent itemset mining: A 25 years review. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 9(6), 2019.

Mahout. Apache software foundation. https://mahout.apache.org//, 2020. [Online; accessed 31-August-2022].

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

RapidMiner. software. https://rapidminer.com/, 2022. [Online; accessed 31-August-2022].

Sebastian Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24), April 2018. doi: 10.21105/joss.00638. URL https://joss.theoj.org/papers/10.21105/joss.00638.

Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data mining: practical machine learning tools and techniques, 3rd Edition*. Morgan Kaufmann, Elsevier, 2011. ISBN 9780123748560. URL https://www.worldcat.org/oclc/262433473.